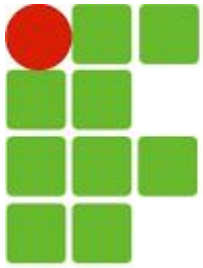


INSTITUTO FEDERAL
GOIÁS
Câmpus Anápolis

Estrutura de Dados I

Sérgio Canuto
Sergio.canuto@ifg.edu.br

Sobre o professor…

- Graduação
  - INF/UFG - Ciência da Computação - 2009
- Mestrado
  - INF/UFG - “Um estudo comparativo entre abordagens supervisionadas para resolução de referências a autores” - 2011
- Doutorado
  - UFMG - “A Thorough Exploitation of Distance-Based Meta-Features for Automated Text Classification” - 2019 (Vencedora CTDBD)
- Publicações:

- Interesses:
  - **Ciência**… em Mineração de Dados, IA, Recuperação de Informações
- Projetos:
  - Classificação de texto, Análise de sentimento em redes sociais, petrobrás, telemarketing…

Sobre o curso…

- Ciência da computação - IFG Câmpus Anápolis:
  - Um dos pouquíssimos cursos de Ciência da Computação **com nota 5** no ENADE em 2023:
    - Único do centro-oeste

CENTRO UNIVERSITÁRIO JORGE AMADO	UNIJORGE
UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE	UFRN
UNIVERSIDADE TUIUTI DO PARANÁ	UTP
UNIVERSIDADE DE PASSO FUNDO	UPF
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOG	IFSP
UNIVERSIDADE DO OESTE PAULISTA	UNOESTE
UNIVERSIDADE FEDERAL DE SÃO PAULO	UNIFESP
UNIVERSIDADE FEDERAL DE VIÇOSA	UFV
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOG	IFG
UNIVERSIDADE FEDERAL DE MINAS GERAIS	UFMG
CENTRO UNIVERSITÁRIO DO INSTITUTO DE EDUCAÇÃO SU	IESB
UNIVERSIDADE FEDERAL DE JUIZ DE FORA	UFJF
Faculdade Cesar	FCE

Sobre os alunos…

- Fora Laboratório de programação/Construção de algoritmos… já teve alguma experiência com desenvolvimento?
- Partindo de suas experiências…
  - Qual a primeira conceituação que vem à mente com o termo “*estrutura de dados*”?

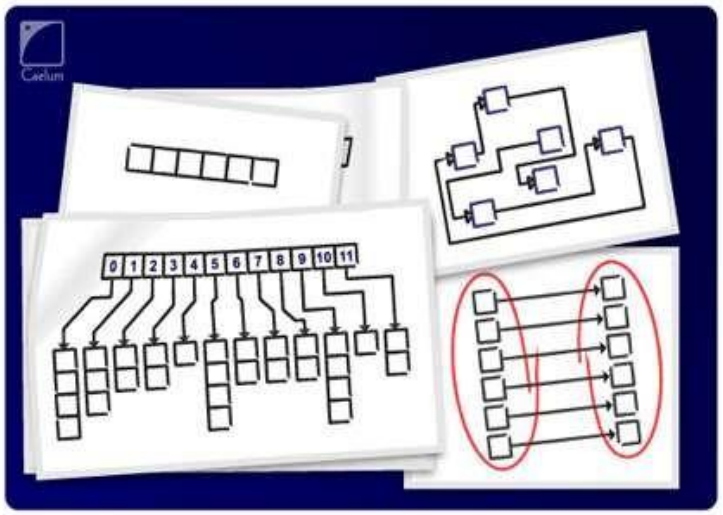
INSTITUTO FEDERAL
Goiás

Sobre os alunos…

- Fora Laboratório de programação/Construção de algoritmos… já teve alguma experiência com desenvolvimento?
- Partindo de suas experiências…
  - Qual a primeira conceituação que vem à mente com o termo “*estrutura de dados*”?
  - Partindo dos seus objetivos com o curso de computação, onde "estrutura de dados" pode contribuir?

Estrutura de Dados

- Programa = Algoritmo + Dados
- Resolução de Problema: abstração
- Cadastro de Clientes
  - Quais dados são importantes?
    - A idade do cliente é importante?
    - A cor do cabelo do cliente é importante?
  - Qual o algoritmo usar?
    - Como encontrar um cliente?
    - Como inserir um novo cliente?



Estrutura de Dados

- “Estrutura de dados é o ramo da computação que estuda os diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento.”

- As estruturas de dados definem a organização, métodos de acesso e opções de processamento para a informação manipulada pelo programa.

- Em geral, algoritmos trabalham sobre **Estruturas de Dados:**
  - Conjunto de dados que representa uma situação real
  - Abstração da realidade

Ementa

- Ponteiros e alocação dinâmica de memória.
- Tipos abstratos de Dados.
  - Listas e suas generalizações:
    - Tipos, operações e implementação.
  - Pilhas e filas:
    - Tipos e implementação.
- Algoritmos de Pesquisa e Ordenação:
  - Bolha, inserção e seleção, quick sort, Merge Sort, Counting Sort, Radix Sort,Bucket Sort.
- Notação e Análise Assintótica.

Objetivos

- Introduzir a análise do custo de execução de algoritmos considerando o tempo e o uso de memória.
- Demonstrar aplicações de estruturas de dados em função do problema.
- Incentivar o autodidatismo.

Habilidades e Competências

- Resolver problemas usando ambientes de programação.
- Aplicar os princípios de gerência, organização e recuperação da informação.
- Reconhecer a importância do pensamento computacional no cotidiano e sua aplicação em circunstâncias apropriadas e em domínios diversos.

Bibliografia

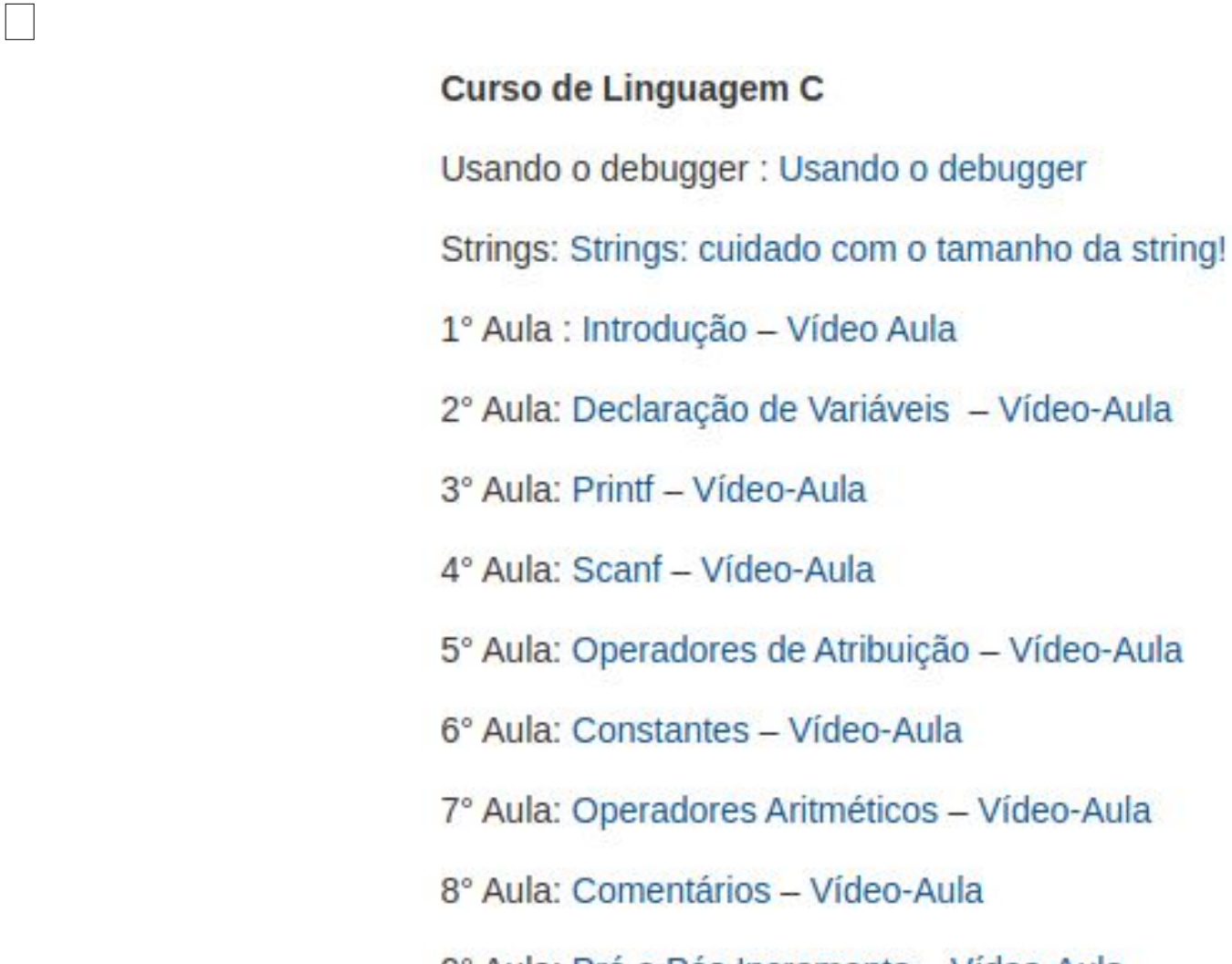
- ANDRÉ BACKES. Estrutura de Dados descomplicada em Linguagem C. Elsevier. (2016).
- THOMAS H. CORMEN, CHARLES E. LEISERSON, RONALD L. RIVEST e CLIFFORD STEIN. Algoritmos:Teoria e Prática. Elsevier. (2012).
- NÍVIO ZIVIANNE. Projeto de Algoritmos: com Implementações em Java e C++. São Paulo: Cengage Learnin.

Bibliografia Complementar

- NÍVIO ZIVIANNE. Projeto de Algoritmos: com Implementações em Pascal e C. São Paulo: Cengage Learnin. (2010)
- ANDRÉ BACKES. Linguagem C completa e descomplicada. Campus.
- NIKLAUS WIRTH. Algoritmos e Estruturas de Dados. LTC. (2012).
- ANA FERNANDA GOMES ASCENCIO. Estrutura de Dados : Algoritmos, Análise da Complexidade e Implementações em Java e C/C++. Pearson Education do Brasil. (2010).
- VILMAR PEDRO VOTRE. C++ Explicado e Aplicado. Alta Books. (2016).

Linguagem C - aulas de implementação

- https://programacaodescomplicada.wordpress.com/indice/linguagem-c/



Curso de Linguagem C

Usando o debugger : Usando o debugger

Strings: Strings: cuidado com o tamanho da string!

1° Aula : Introdução – Vídeo Aula

2° Aula: Declaração de Variáveis – Vídeo-Aula

3° Aula: Printf – Vídeo-Aula

4° Aula: Scanf – Vídeo-Aula

5° Aula: Operadores de Atribuição – Vídeo-Aula

6° Aula: Constantes – Vídeo-Aula

7° Aula: Operadores Aritméticos – Vídeo-Aula

8° Aula: Comentários – Vídeo-Aula

Estrutura de dados - aulas de implementação

- https://programacaodescomplicada.wordpress.com/indice/estrutura-de-dados/

Curso de Estrutura de Dados

1° Aula: Tipo Abstrato de Dados(TAD) – Vídeo-Aula

2° Aula: Modularização – Vídeo-Aula

3° Aula: Listas pt.1 – Definição – Víde0-Aula

4° Aula: Listas pt.2 – Lista Estática Sequencial – Vídeo-Aula

5° Aula: Listas pt.3 – Implementação de Lista Estática – Vídeo-Aula

6° Aula: Listas pt.4 – Informações da Lista Estática – Vídeo-Aula

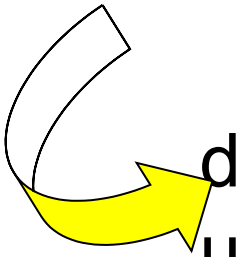
7° Aula: Listas pt.5 – Inserção na Lista Estática – Vídeo-Aula

Avaliação

- Prova 1 (40%)
  - 06/10
- Prova 2 (40%)
  - 01/12
- Participação & Avaliação continuada (20%)
  - Atividades e exercícios de implementação.

Tipos de Dados

Tipo de dado



definição do conjunto de valores (domínio) que uma variável pode assumir

Ex: inteiro

< … -2, -1, 0, +1, +2, … >

lógico

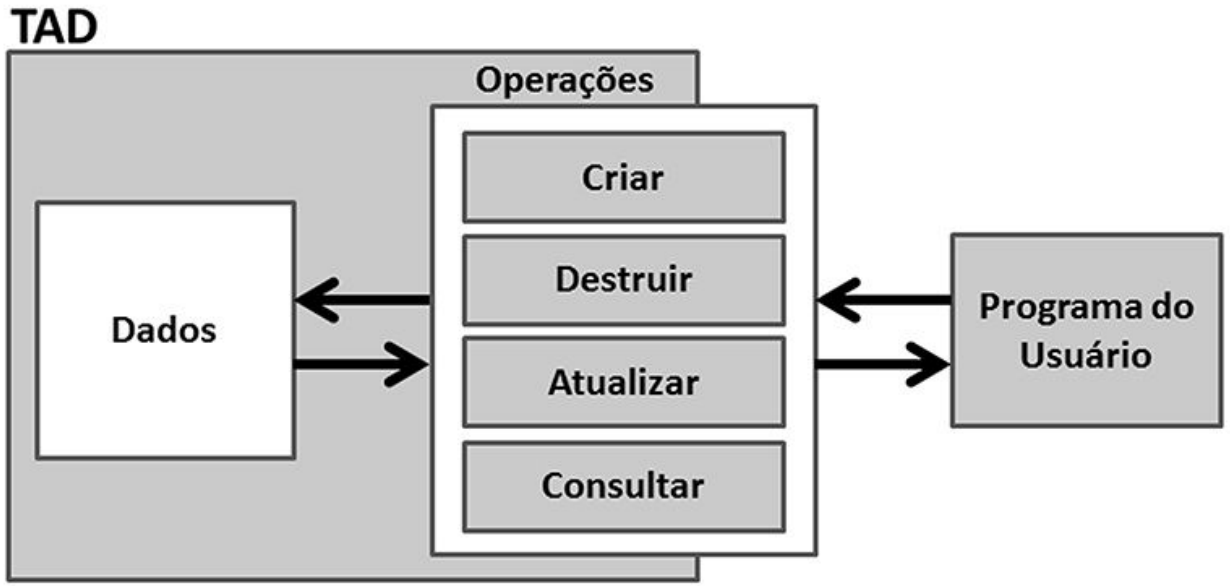
< verdadeiro, falso >

Tipos de Dados

- **Tipos básicos (primitivos)**
  - inteiro, real, e caractere
- **Tipos de estruturados (construídos)**
  - arranjos (vetores e matrizes)
  - seqüências (conjuntos)
  - referências (ponteiros)
- **Tipos definidos pelo programador**

Tipo Abstrato de Dados:
definidos pelo programador

- Um **tipo abstrato de dados**, ou **TAD**, é um conjunto de dados estruturados e as operações que podem ser executadas sobre esses dados.
- Tanto a **representação** quanto **as operações** do **TAD** são especificadas pelo programador.
  - O usuário utiliza o **TAD** como uma caixa-preta por meio de sua **interface**.

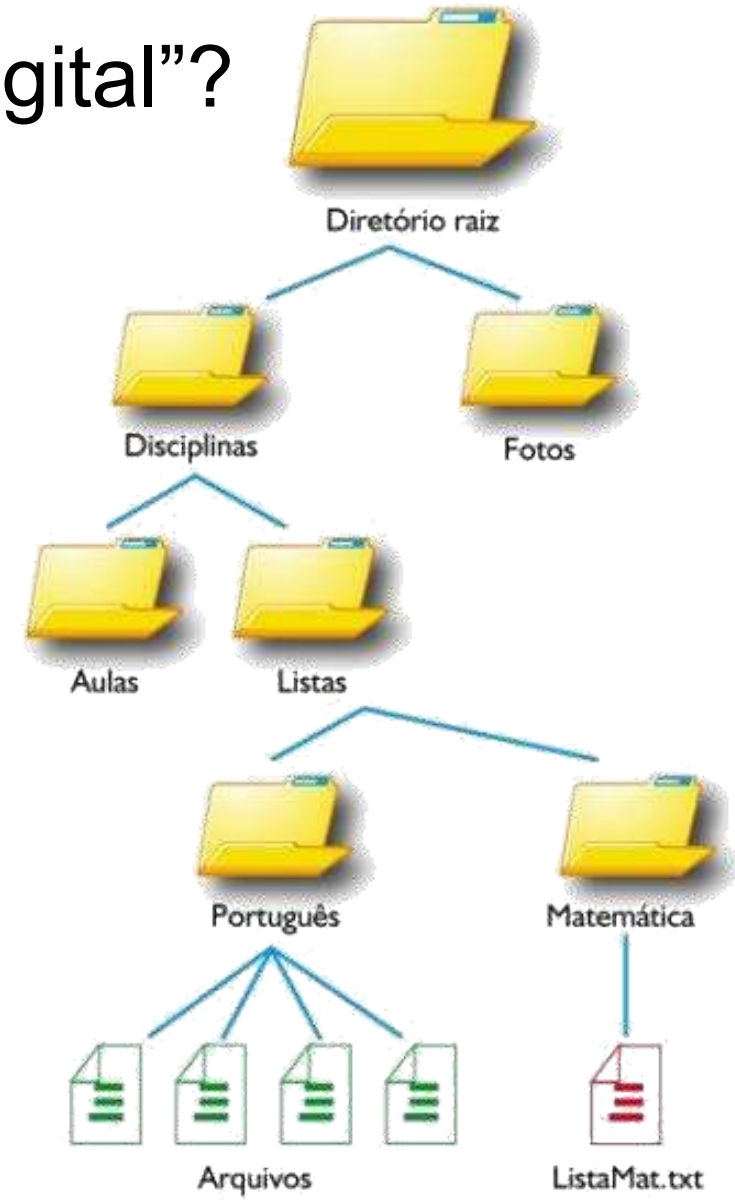


Tipos e Estruturas de Dados

- Tipos de dados básicos
  - Fornecidos pela Linguagem de Programação
- Estruturas de Dados
  - Estruturação *conceitual* dos dados
  - Reflete um **relacionamento lógico** entre dados, de acordo com o problema considerado

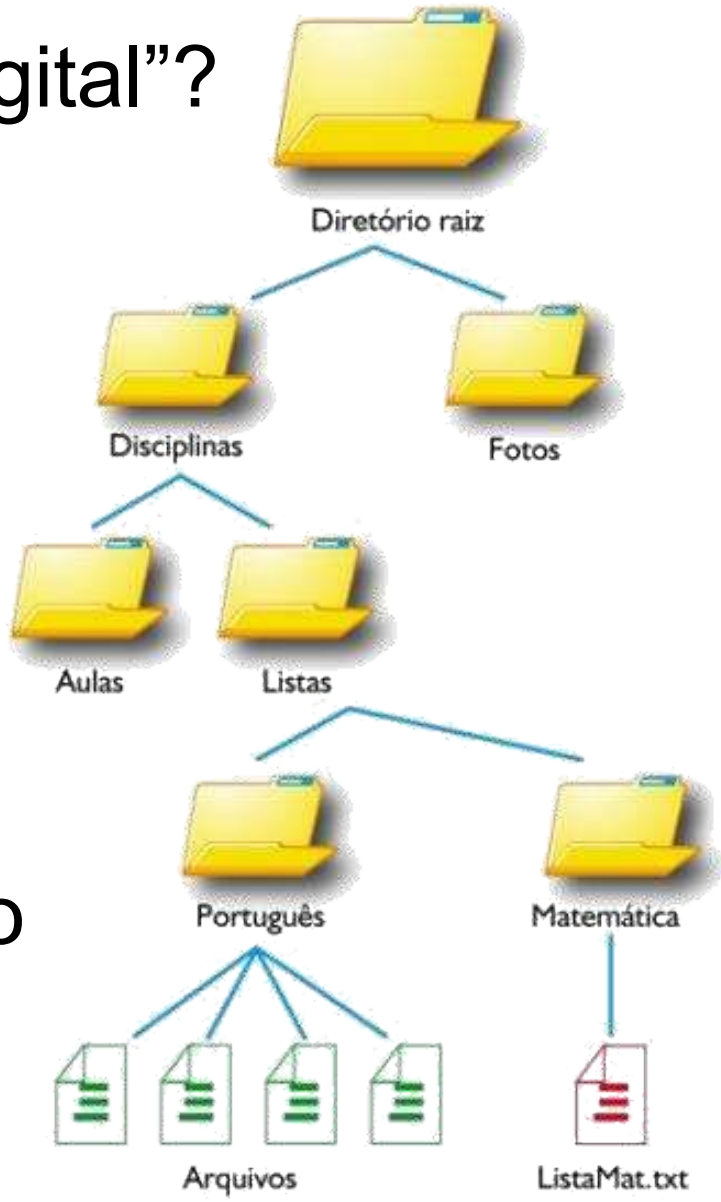
Estrutura de Dados

- O que é um “dado digital”?
- O que o diferencia de “lixo digital”?



Estrutura de Dados

- O que é um “dado digital”?
- O que o diferencia de “lixo digital”?
- Sua **organização**
  - Sabemos como encontrá-los
- E isso permite...
  - Busca
  - Remoção
  - Inserção...
- Organização → Desempenho



Estrutura de Dados no Dia-a-Dia

- Representar a organização de uma empresa
  - 1 presidente, 1 vice-presidente, 1 diretor de vendas e 1 de criação, este último com 2 subdiretores?

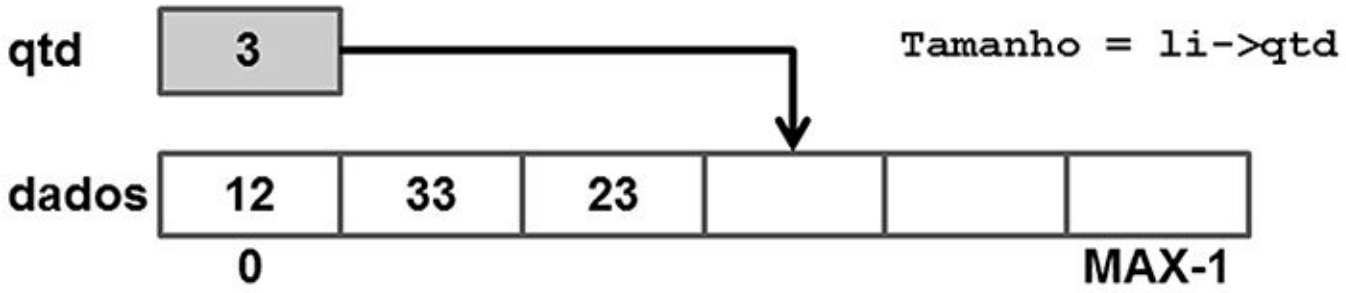
Estrutura de Dados no Dia-a-Dia

- Como representar a bibliografia do curso?
  - Algoritmos:Teoria e Prática
  - Estrutura de Dados descomplicada em Linguagem C
  - Projeto de Algoritmos: com Implementações em Java e C++

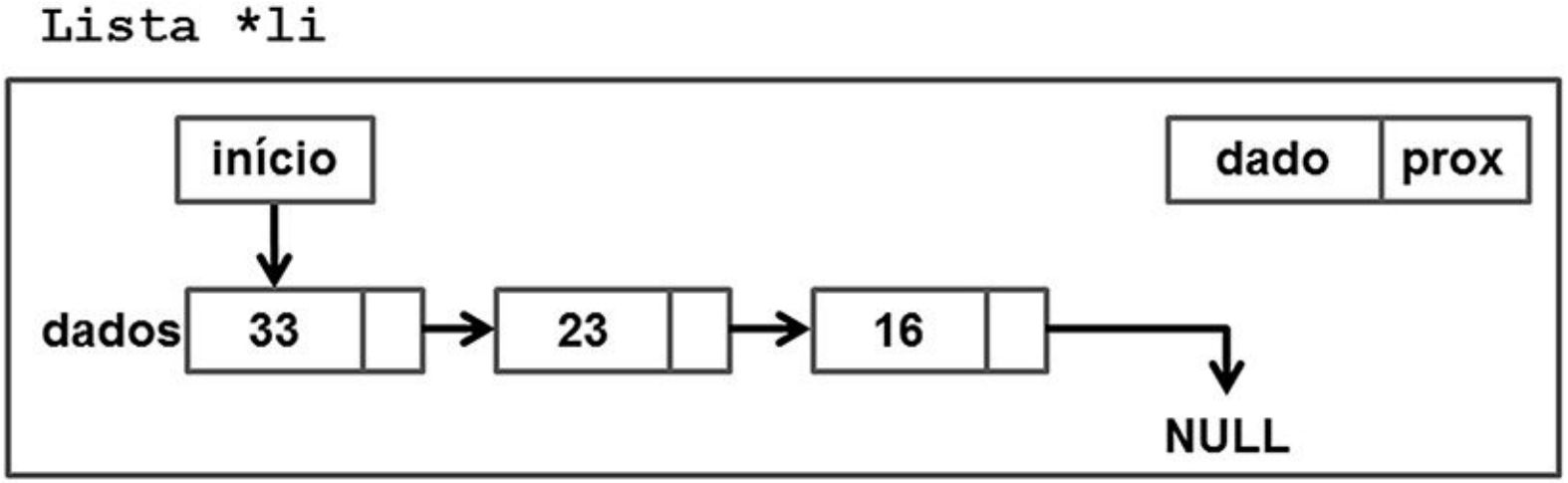
Lista

Estrutura de Dados no Dia-a-Dia

- Como implementar uma lista?
  - vetores:



- Encadeamento:



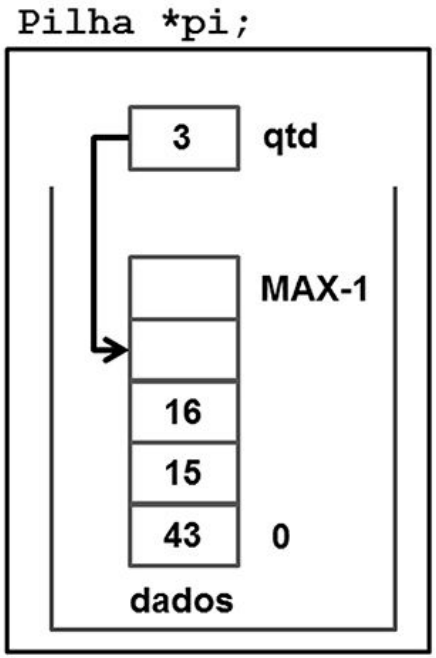
Estrutura de Dados no Dia-a-Dia

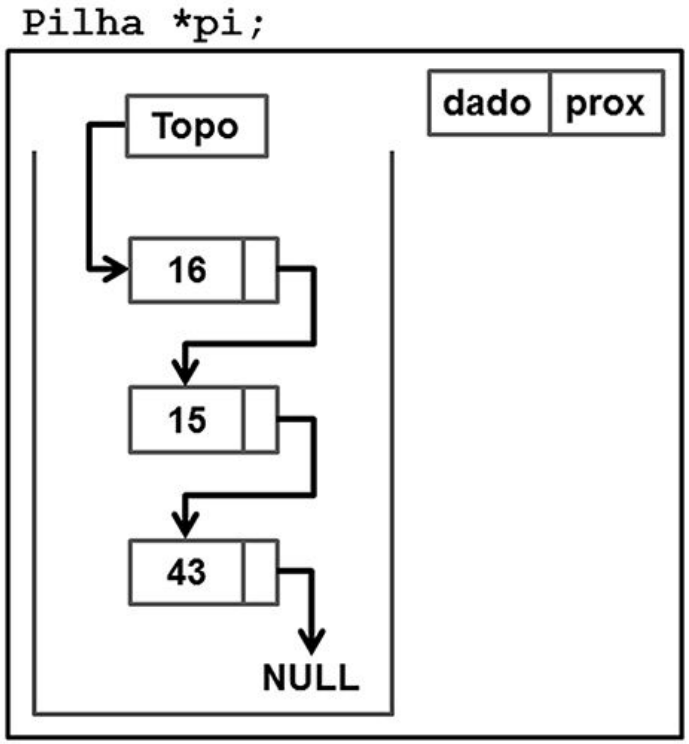
- Como o motoboy organiza as pizzas?

Pilha

Estrutura de Dados no Dia-a-Dia

- Como implementar pilhas?
  - Vetores, encadeamentos (só precisa alterar/pegar o início)





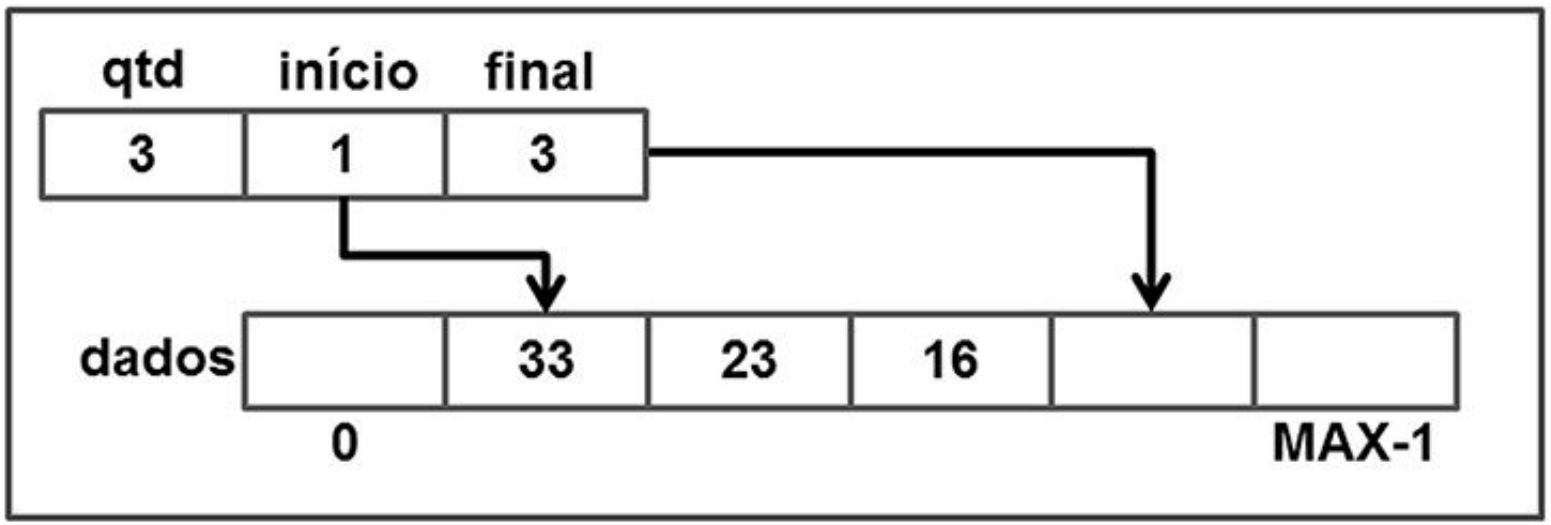
Estrutura de Dados no Dia-a-Dia

- Como as pessoas esperam no banco?

Estrutura de Dados no Dia-a-Dia

- Como implementar uma fila?
  - Precisa ter um controle do início (para obter/remover o dado) e do final da fila (para inserir)

```
Fila *fi;
```



Pesquisa e Ordenação dos Dados

Pesquisa e Ordenação dos Dados

- Conjunto Ordenado:
  - Elementos dispostos sob uma determinada ordem.

- Objetivos da Ordenação:
  - Facilitar a recuperação;
  - Tornar mais eficiente o acesso aos dados.

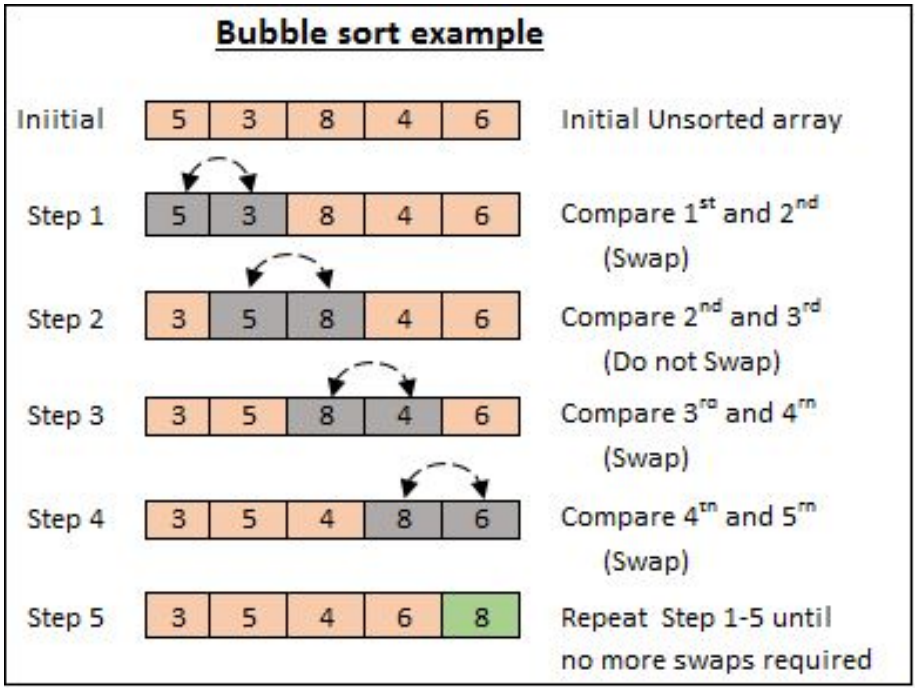
- Visão crítica da importância de bons algoritmos.

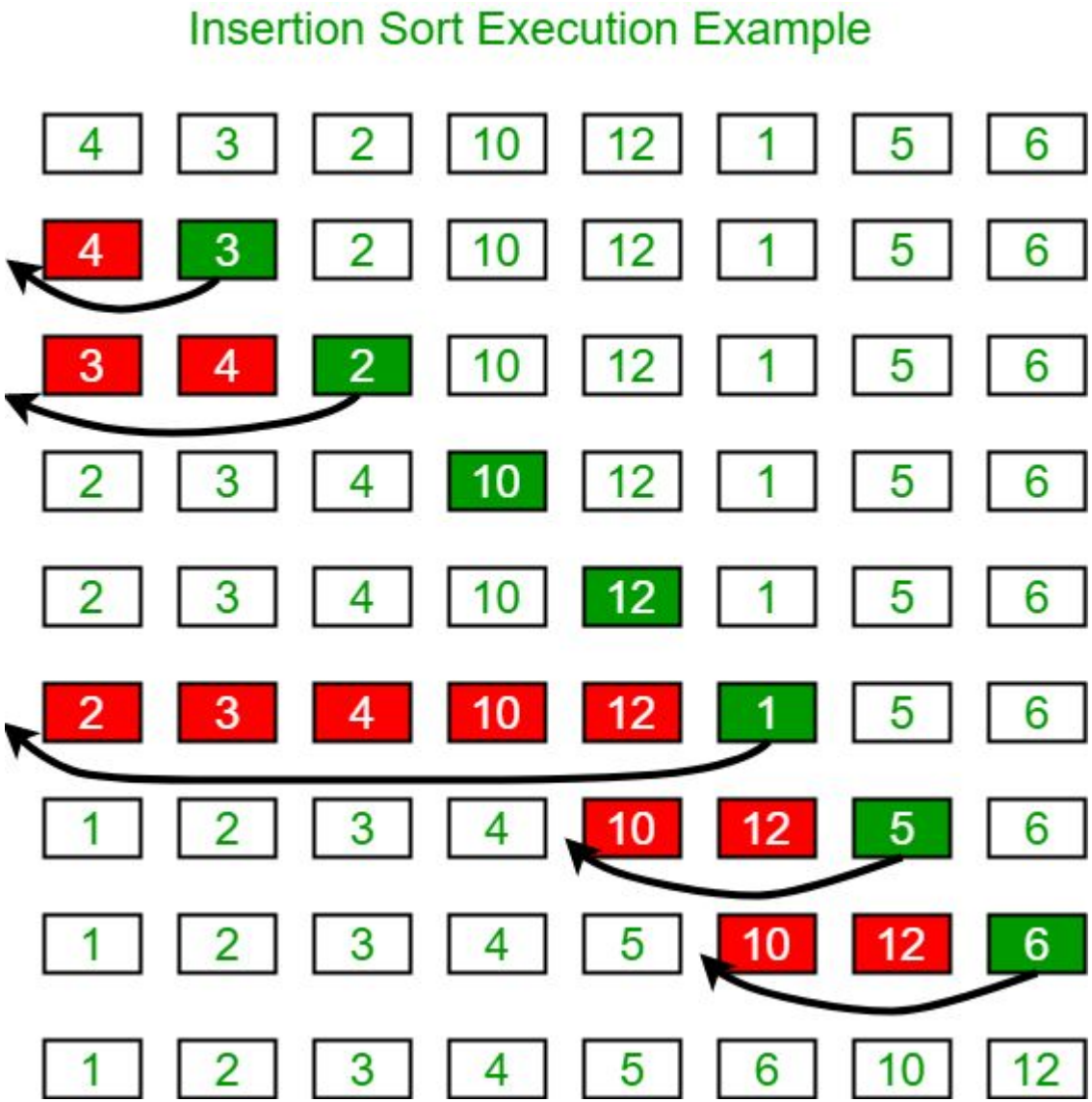
Pesquisa e Ordenação dos Dados

- Tipos de Ordenação:
  - Bolha
    - https://www.youtube.com/watch?v=nmhjrI-aW5o
  - Inserção
    - https://www.youtube.com/watch?v=OGzPmgsI-pQ
  - Seleção
    - https://www.youtube.com/watch?v=xWBP4lzkoyM
  - 15 algoritmos:
    - https://www.youtube.com/watch?v=kPRA0W1kECg

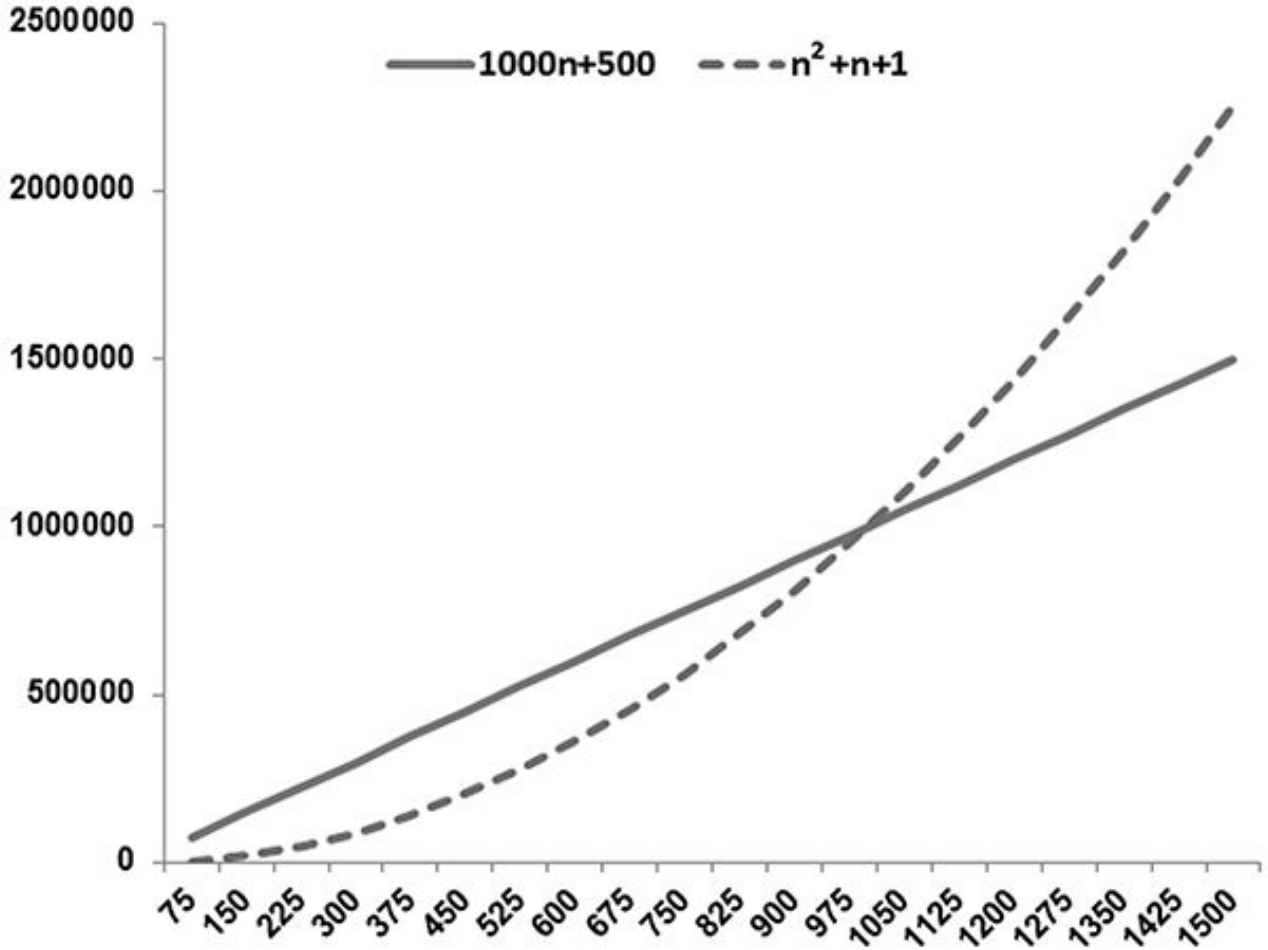
- Qual deles é mais “rápido”?
  - Análise Assintótica

Pesquisa e Ordenação dos Dados





Análise assintótica



Implementação

Linguagem C

- Acesso direto à memória
  - Controle de ponteiros.
  - Controle da alocação de recursos.
- Uma das mais bem sucedidas linguagens de programação.
- Base para diversas linguagens (C++, C#, Java, etc)
- Uma das linguagens mais utilizadas no mundo

EXEMPLO – PROGRAMA.C

```
#include<stdio.h>
#define MAX 10
int LeInteiro() {
   int num;
   printf("Digite um numero: ");
   scanf("%d", &num);
   printf("\n");
   return num;
}
```

```
int main() {
   int v[10], cont, aux, i, soma;
   float m;
   cont = 0;
   aux = LeInteiro();
   while(aux >= 0) {
     v[cont] = aux;
     aux = LeInteiro();
     cont++;
   }
   soma = 0;
   for(i=0;i<cont;i++)
     soma += v[i];
   m = soma / (float) cont;
   printf("resultado: %f\n",m);
}
```

- Compilador:

  gcc -o programa.exe programa.c

Alocação de Memória

- Alocação estática:
  - A quantidade de memória necessária para armazenar as suas variáveis é automaticamente reservada na **stack** (pilha)
  - O programador também não tem controle sobre o tempo de vida dessas variáveis na memória.
    - A **stack** guarda os dados alocados dentro dos escopos de funções

Alocação de Memória - A Variável

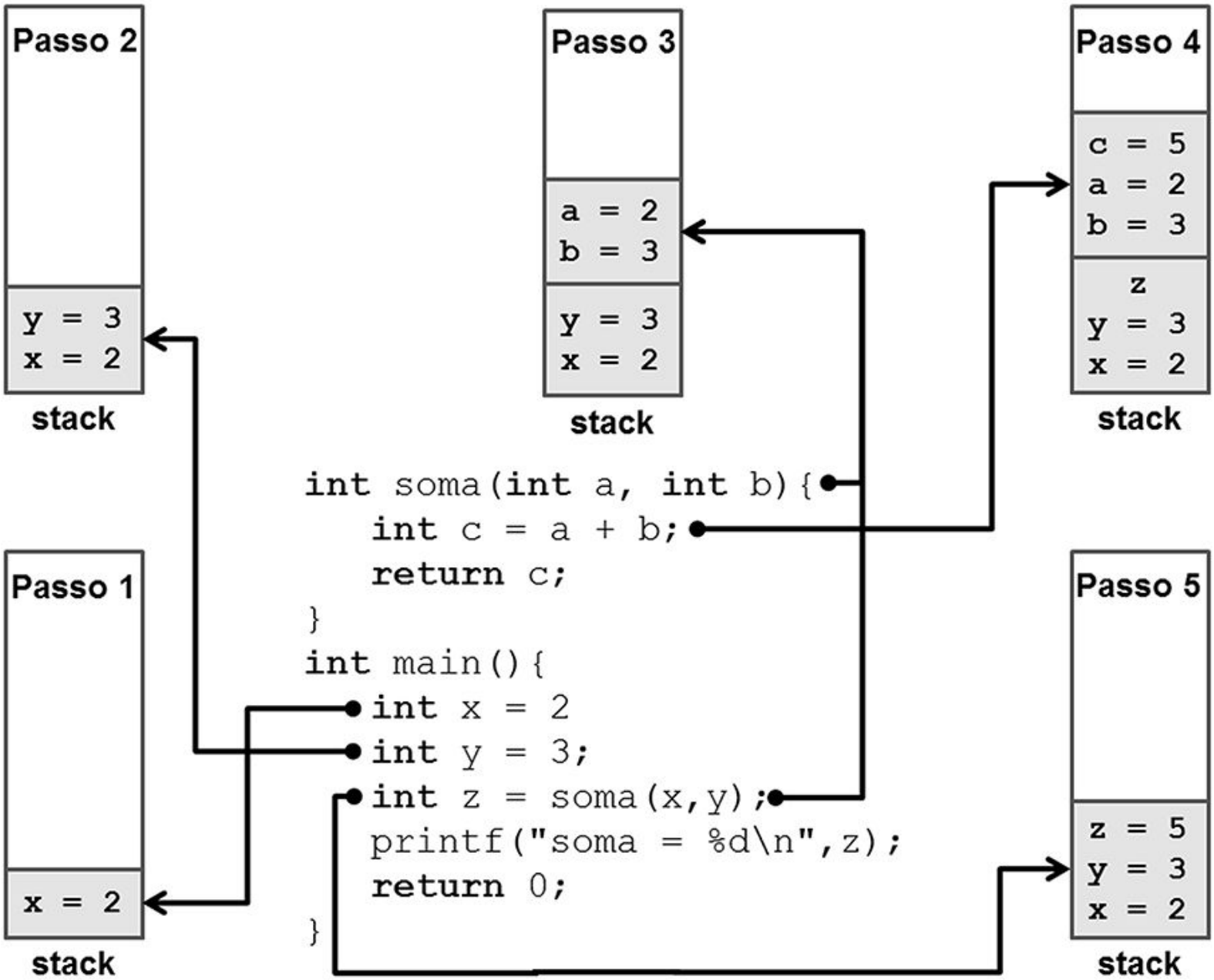
– int idade = 31;

- Nome da variável: idade
- Tipo da variável: int
- Conteúdo da variável: 31
- Endereço da variável: 1

Alocação de Memória

- Alocação estática:
  - As variáveis são empilhadas na stack à medida que são declaradas na main() (Passos 1 e 2).
  - Ao correr uma chamada de função, cria-se uma nova região na stack com os parâmetros locais (Passo 3).
  - Variáveis declaradas dentro da função são empilhadas dentro da sua região da stack (Passo 4).
  - Terminada a execução da função ela é removida da stack (desempilhada) e seu valor retornado para uma variável dentro da main(). A variável declarada para receber o retorno da função é empilhada dentro da região da stack associada a main() (Passo 5).As variáveis são empilhadas na stack à medida que são declaradas

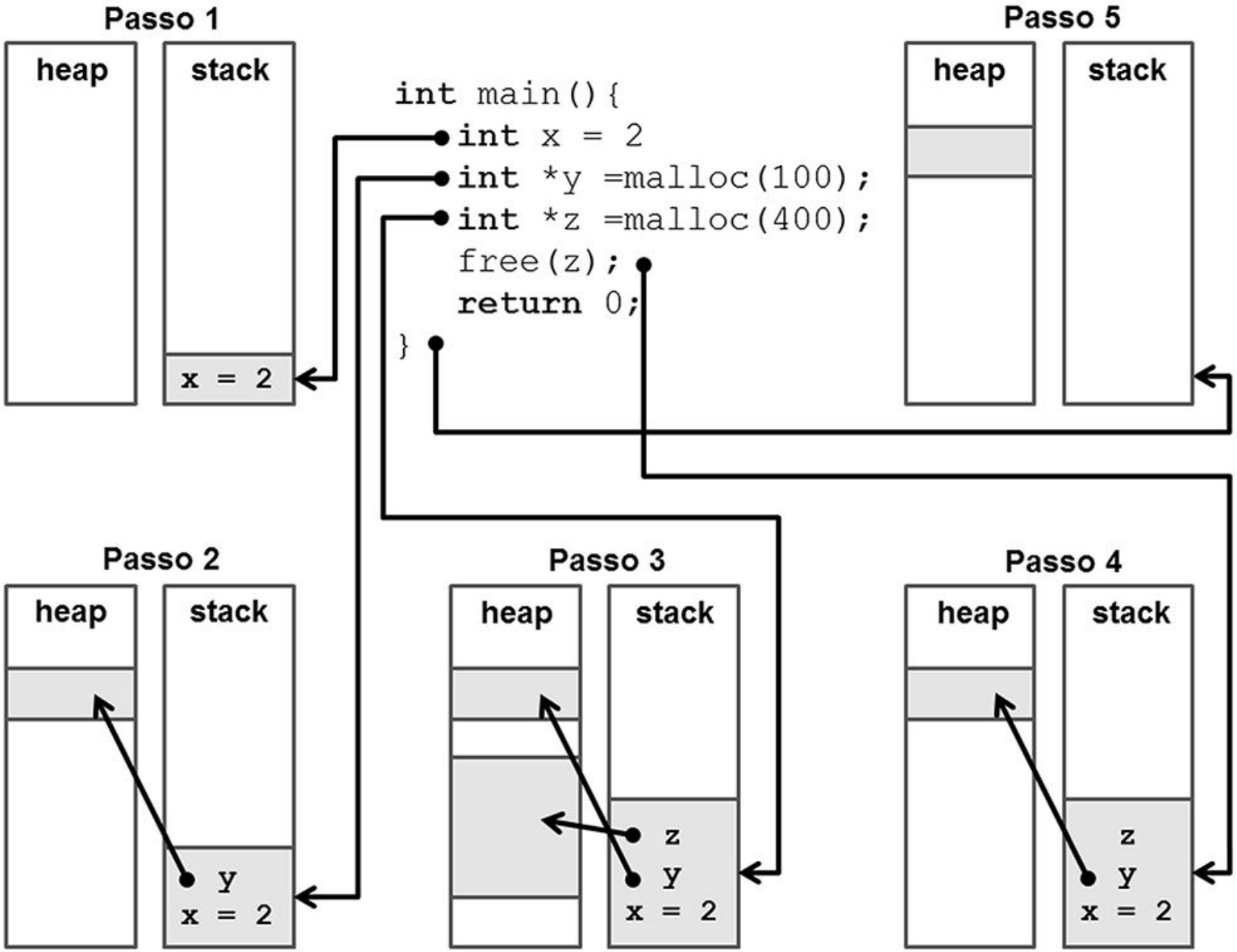
Alocação de Memória Estática



Alocação de Memória

- Alocação dinâmica:
  - A quantidade de memória necessária para armazenar as suas variáveis é **manualmente** reservada na **heap**
  - O programador é responsável por reservar/liberar a quantidade de memória necessária para seus dados.
    - Para acessar a **heap**, os dados são alocados dinamicamente por meio da função **malloc()** em C, e só podem ser acessados por ponteiros.

Alocação de Memória Dinâmica

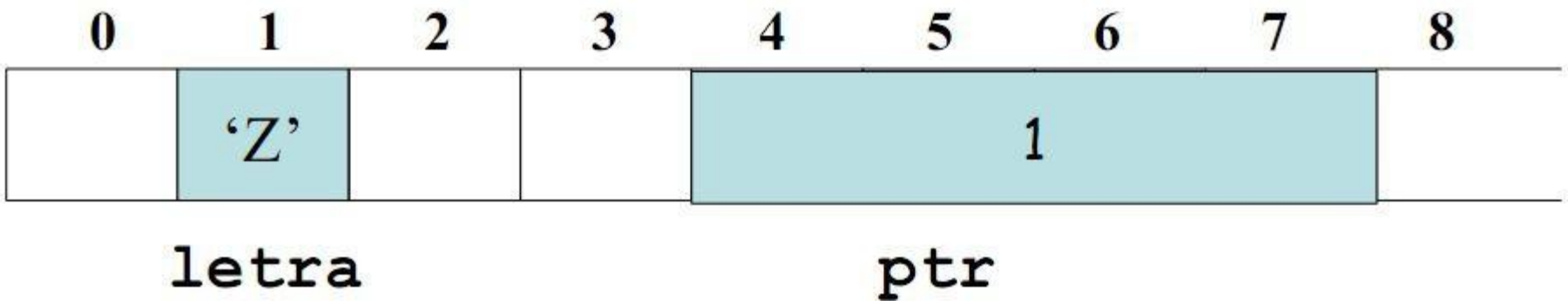


Alocação de Memória Dinâmica:
Variáveis do tipo ponteiro

char letra = ‘Z’;

char *ptr = &letra;

- Nome da variável: ptr
- Tipo da variável: char *
- Conteúdo da variável: 1
- Endereço da variável: 4



Alocação Estática vs Dinâmica

Alocação estática (stack)	**Alocação dinâmica (heap)**
Armazenado na memória RAM	Armazenado na memória RAM
Variáveis são liberadas automaticamente no final do escopo	Variáveis não dependem do escopo (podem ser acessadas globalmente) e devem ser liberadas manualmente
Alocação mais rápida que na heap	Alocação mais lenta que na stack
Implementado usando uma estrutura de dados do tipo pilha	Blocos de dados são alocados sob demanda
Armazena dados locais e endereços de retorno utilizados na passagem de parâmetros	Pode sofrer fragmentação após sucessivas alocações e liberações de memória
Os dados podem ser usados sem ponteiros	Os dados são acessados por ponteiros
Pode ocorrer estouro de pilha quando a pilha é muito usada	A alocação pode falhar se muita memória é solicitada
É usada quando se sabe exatamente o quanto de espaço será alocado antes do tempo de compilação e esse espaço não é muito grande	É usada quando não se sabe exatamente o quanto de espaço será alocado antes do tempo de compilação ou esse espaço é muito grande
Geralmente possui um tamanho máximo predeterminado quando o programa inicia	Responsável por vazamentos de memória